

# CrowdSurf: Empowering Informed Choices in the Web<sup>\*</sup>

Hassan Metwalley<sup>1</sup>, Stefano Traverso<sup>1</sup>, Marco Mellia<sup>1</sup>,  
Stanislav Miskovic<sup>2</sup>, and Mario Baldi<sup>1,2</sup>

<sup>1</sup> Politecnico di Torino, Italy

{metwalley, traverso, mellia}@tlc.polito.it

<sup>2</sup> Symantec Corp., USA

{stanislav\_miskovic, mario\_baldi}@symantec.com

**Abstract.** When surfing the Internet, individuals leak personal and corporate information to third parties whose (legitimate or not) businesses revolve around the value of collected data. The implications are serious, from a person unwillingly exposing private information to an unknown third party, to a company unable to manage the flow of its information to the outside world. The point is that individuals and companies are more and more kept out of the loop when it comes to control private data.

With the goal of empowering informed choices in information leakage through the Internet, we propose CROWDSURF, a system for comprehensive and collaborative auditing of data that flows to Internet services. Similarly to open-source efforts, we enable users to contribute in building awareness and control over privacy and communication vulnerabilities. CROWDSURF provides the core infrastructure and algorithms to let individuals and enterprises regain control on the information exposed on the web.

We advocate CROWDSURF as a data processing layer positioned right below HTTP in the host protocol stack. This enables the inspection of clear-text data even when HTTPS is deployed and the application of processing rules that are customizable to fit any need. Preliminary results obtained executing a prototype implementation on ISP traffic traces demonstrate the feasibility of CROWDSURF.

## 1 Introduction

Users increasingly rely on Internet services, looking for news and products, accessing social networks, organizing their life, etc. There are companies that base their business on the collection of personal information implicitly or explicitly embedded in the above users' activities. This results in leakage of information that users and companies prefer to keep private, in people being exposed to dubious third-party services, as well as in web companies (sometimes illegitimately) tracking their users. This phenomenon is ubiquitous, with even the major players taking part in it [1,2,10].

Hence, users' concerns about privacy and information leakage largely increased, motivated also by recently exposed government surveillance programs. However, no means exist to control which data is handed to the web.

To this end, a common misconception is that encryption would solve the problem. Accordingly, HTTPS usage increased by 100% each year, reaching about 42% of web

---

<sup>\*</sup> This work was conducted under the Narus Fellow Research Program.

flows in June 2014 [11]. In reality, the effects are quite different and rather exacerbate the problem. Firstly, encryption increases the value of data. Specifically, web services that deploy encryption establish a monopoly on information by precluding any other parties from deploying it, thus gaining a huge advantage in today's Internet where many businesses revolve around user information. Secondly, when HTTPS is deployed, users have no chance to rely on third parties to check and possibly choose which (personal) information they are sharing.

In this scenario, we advocate the need of a communication model where users are explicitly offered the freedom to i) understand which services get their data, and ii) govern which information they are asked to exchange. We envision a holistic and flexible solution to verify and control the information which is exchanged on the Internet, when using a web browser or running a smartphone app, whether connected to the corporate network or to a public WiFi hotspot.

CROWDSURF, presented in this paper, provides a framework for such a solution. It is designed as an open service to which anyone can easily contribute. A part of CROWDSURF resides on client devices to both provide visibility on traffic and possibly act upon it (e.g., by modifying or blocking information). We conceived CROWDSURF as a new layer that sits right between the application and the protocol stack, where information has not yet been encrypted. CROWDSURF targets web surfing, and thus HTTP, the new “narrow waist of the Internet” [13]. This empowers the protection of the user's data and optional contributions to the system by users themselves. Anyone contributes according to a personal level of expertise or convenience, from teams of security researchers who can collaboratively measure intricate signs of behind-the-scenes communications between the service providers, to novice users who can simply offer anonymized samples of their traffic or vote on the legitimacy of data leaving their devices.

Another part of CROWDSURF resides on open cloud servers performing intensive processing tasks over massive datasets obtained through the contribution of volunteers. Specifically, the cloud component runs algorithms that clean and rank users' voting, index voluntarily submitted traffic, and attempts to discover unknown types of information leakage. All of the data gathered and processed enables the cloud CROWDSURF component to compute pieces of *advice* about the trustfulness of web services. This advice is shared with all the resident components that can leverage it to support users in taking informed decisions. Users can create *rules* based on the received advice to enable fine grained control on the information flow. For instance, users can choose to block undesired services, or filter private information, or explicitly embrace third-party services.

The technology offered by CROWDSURF is essential not only for individual users, but also for companies with the need to control the information entering and leaving the corporate network. Currently, companies are forced to trust the devices connected to the network and have hard time verifying the information they exchange. The so called “BYOD” (Bring Your Own Device) phenomenon and the reduced efficacy of traditional approaches based on firewalls and IDS's (mined by encryption [11]) further exacerbate the problem. In the corporate scenario, the open cloud service is replaced

by a private component that, through the resident components, can impose filters to any device connected to the corporate network.

At last, CROWDSURF allows third-parties to offer novel services, possibly complementing current client-server-based ones. For instance, CROWDSURF could be used to enable a user to voluntarily use an accelerating proxy offered by an ISP only for specific types of traffic (e.g., when watching videos, but not when accessing her bank account). CROWDSURF would dynamically forward traffic to the proxy or directly to the final destination depending on a set of rules provided by the user or by a third party and relying on advice obtained by the system. CROWDSURF could even be instrumental in enabling users to monetize on their personal information, should they decide for it, as proposed in [14].

## 2 CROWDSURF Description

We envision a crowd-sourced system in which users can voluntarily opt-to collaborate by providing explicit (e.g., their opinion) and implicit (e.g., traffic samples) information on the web services they use. In return they obtain information about web services. A *collector* – running in the cloud – collects information provided by the users, and it feeds an automatic *data analyzer*, which runs data mining algorithms to produce *advices*. An advice contains indications about trustfulness of web services. For instance, the data analyzer can flag services collecting/leaking users’ personal information, or services that children should not access, or services known to host malicious software. A federated group of experts, the *advising community*, inspects the results provided by the data analyzer and interacts with it to generate the advices. Following a collaborative approach similar to Wikipedia and the Electronic Frontier Foundation<sup>3</sup> (EFF), users are invited to increase the system’s “wisdom”. They can be active in controlling the personal information they expose to services, or in forming the advices. And then they can voluntarily donate portions of their browsing activity, i.e., anonymized HTTP-level traces. The advising community is supported by data mining algorithms that automatically raise flags.

The Internet offers some tools to help users to avoid disclosing personal information when browsing the web, e.g., popular browser plugins such as DoNotTrackMe<sup>4</sup>, EFF’s Privacy Badger<sup>5</sup>, WoT<sup>6</sup> and Ghostery<sup>7</sup>. For mobile terminals, some proposals offer similar ideas [5,8]. Each targets some specific aspects of privacy leakage only. Some leverage the idea of a crowd-sourced approach to inform users about website trustfulness. More holistic technologies as TOR [7] offer protection to users identity and from traffic inspection attacks, but they do not curb the personal information which is exposed to servers at application layer. Similarly, companies are offering solutions to control web browsing [3]. Yet, what they offer is unknown, and mostly un-verifiable. CROWDSURF is all of them, and none of them. CROWDSURF’s challenge is in offering a

<sup>3</sup> <https://www.eff.org/>

<sup>4</sup> <http://www.abine.com/donottrackme.html>

<sup>5</sup> <https://www.eff.org/privacybadger>

<sup>6</sup> <https://www.mywot.com/>

<sup>7</sup> <https://www.ghostery.com/en/>

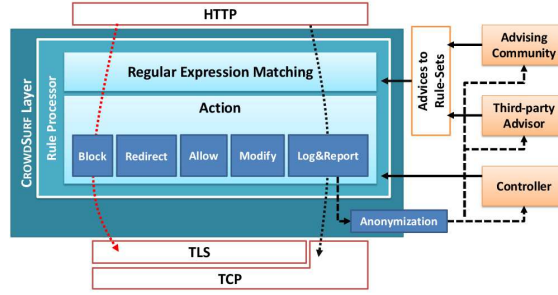


Fig. 1: CROWDSURF layer in the Internet stack and its high level structure.

unified system to overcome limitation of current systems, which often do not cooperate and are dominated by manual decisions. We propose a flexible system that, based on the knowledge of the crowd and supported by automated algorithms, empowers users and companies, offering them the chance to retake the control on private data.

## 2.1 CROWDSURF System Design

For the design of CROWDSURF we follow a short list of simple design requirements:

- 1) **Crowd-sourced**: we want the system to engage users to improve its effectiveness.
- 2) **Anonymity**: we want CROWDSURF to never offend users' privacy. Hence any contribution must be purged from any piece of personal information.
- 3) **Automated**: the system have to automatically process users' contributions to generate the advices.
- 4) **Client centric**: CROWDSURF must be available on any device, as the default tool to support users' choices.
- 5) **Easy to use**: we want CROWDSURF to be as simple and automatic as possible to allow anyone to use it.

Given these principles, we imagine CROWDSURF cornerstone as a new layer to add to the Internet stack. We expect users' terminals, mobiles and personal computers alike, to embed the CROWDSURF layer in their operating system. Fig. 1 represents the high level architecture of the CROWDSURF layer. It sits between the HTTP and the transport layer, where it handles HTTP traffic, before it is eventually encrypted. This choice is motivated by the fact the today HTTP is "the" application layer [13].

Users asynchronously obtain advices from the advising community, and they are free to decide to what level to take them into account: users are free to accept or over-rule the notification of a potential danger. The system implements this feature through the Advices to Rule-Sets block in Fig. 1. It enables controlling how advices are translated in a set of *rules*, or *rule-set*. A rule consists of a *regular expression* and one or more *actions*. For each HTTP request, the Rule Processor looks for matches and applies the corresponding action, for instance Block, Redirect, Modify, Log&Report, etc., with Allow being the default one. This simple pattern matching/action process has proved very flexible and very efficient. It is at the root of successful technologies such as the one used in firewalls, antiviruses, traffic classifiers, etc.

Given that CROWDSURF aims at supporting a crowd-sourced approach, the Log&Report block is vital. It enables the collection of data samples before traffic is possibly encrypted. The layer can perform measurements at user's will and under user's control.

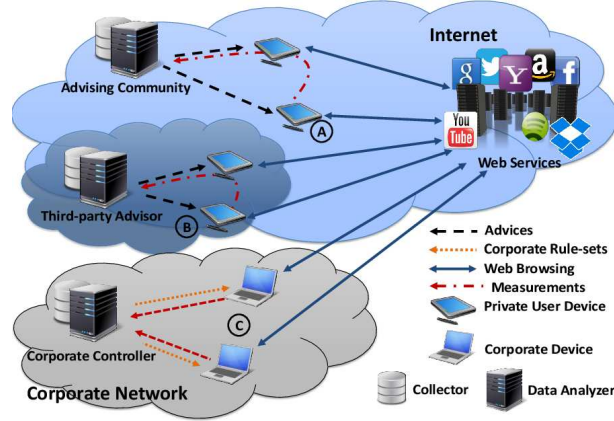


Fig. 2: The entities participating to CROWDSURF and their deployment for different network scenarios.

The layer temporarily stores the measurement data locally until a certain amount is reached, at which point the layer transmits the data to the collector. Since protecting user's privacy is strategic for CROWDSURF, we adopt different approaches to avoid compromising it. The anonymization block in Fig. 1 is responsible for this. First, it implements sampling policies, e.g., by logging only a fraction of traffic at random. This also reduces the amount of data to transfer. Second, it filters out any piece of personal information. E.g., by default, all key values are replaced by random strings by using cryptographic hash functions. Then, a pattern/action mechanism is used. As before, the community can supply pre-defined lists of anonymization practices, which can always be customized by the user. For instance a generic policy "remove all possible password fields" can be augmented with "never collect data when browsing my online bank account". Third, each user is assigned a unique random identifier, rotated periodically (e.g., every day). Fourth, data on the collector will be stored for only the time needed to process it (also to limit the storage at the collector). At last, since even information available at the network layer (e.g., IP addresses) could be exploited to trace back the identity of the user, transmission to the collector takes place through an encrypted channel established over a randomly chosen CrowdProxy, i.e., by employing other devices running the CROWDSURF layer as relays. The collector automatically provides the identity of other CROWDSURF devices among which to randomly choose the relay.

Fig. 2 represents possible deployment scenarios. Private user A accessing the Internet receives advices from the advising community (dashed black unidirectional arrows) and possibly use them to regulate its access to web services (solid blue double-headed arrows). If A's preferences allow it, traffic samples are sent to the collector via a CrowdProxy (dashed red arrows).

The advising community and public data analyzer may be supported by public bodies or non-profit organization like EFF. However, advices could also be generated by a third-party advisor run by an independent, third-party entity which offers custom advices to users. This opens a "market of advices". For instance, user B opts for a service offered by a third party advisor.

		Actions		
		block	redirect	log&report
Web Services	Facebook	C		
	Twitter			C
	Dropbox			C
	Google		C( $\rightarrow$ Bing)	
	YouTube	C		
	Ebay+Amazon	C		
	Adult Sites	C, K		
	Trackers	P		K
	Ads+NoJS	P		

Table 1: Rules for the (P)aranoid, (K)id and (C)orporate profiles.

Finally, as shown in the bottom half of Fig. 2, CROWDSURF can also be deployed in a corporate scenario. In this case the corporate controller does not create advices, but it directly imposes rule-sets (orange dotted arrows) which are installed on devices connected to the corporate network (employee C). Indeed, we expect the employee not to be allowed to modify the rule-sets imposed by the corporate authority. Notice also that devices may be asked to report employees’ browsing activity on administrator’s demand directly to the corporate controller, without involving other devices. The presence of the corporate controller must be automatically identified by any device connected to the corporate network including those BYOD. This can be achieved for instance using DHCP extensions, or using standard DNS names that forces the CROWDSURF layer to connect to the corporate controller. Notice that the same rule can be imposed on any corporate-owned device even when connected from other networks.

## 2.2 CROWDSURF Application Examples

In the following we describe examples of CROWDSURF applications in both the public Internet, and the corporate network. We use the same examples to run the experiments presented in Sec. 3.1.

A summary of rules is available in Tab. 1. We define a “Paranoid Profile” that opts for blocking all advertisement sites, to not run Javascript code, and to use private navigation mode on the browser. This profile is the equivalent of running AdBlockPlus and NoScript plugins. This user decides to not share any traffic samples with the community.

A second profile is called “Kid Profile”: the user activates parental control by installing the advices provided by the advising community. In the experiment, we simply use the list of the Alexa top 50 “Adult Sites” augmented by other manually verified adult sites. The user contributes also to manually signal other offending websites/objects he gets into. Finally, he volunteers to enable logging and reporting of the three most popular online trackers (*doubleclick.net*, *scorecardresearch.com*, and *yieldmanager.com*).

A third profile impersonate the “Corporate Profile”: rules are imposed by the network administrator, and i) do not allow employees to access Facebook (also removing Facebook buttons from any website), ii) redirect all requests from Google search to Bing search, iii) block the usage of adult sites, Ebay, Amazon, and YouTube, and iv) all HTTP(S) requests exchanged with Dropbox and Twitter are reported to the corporate collector.

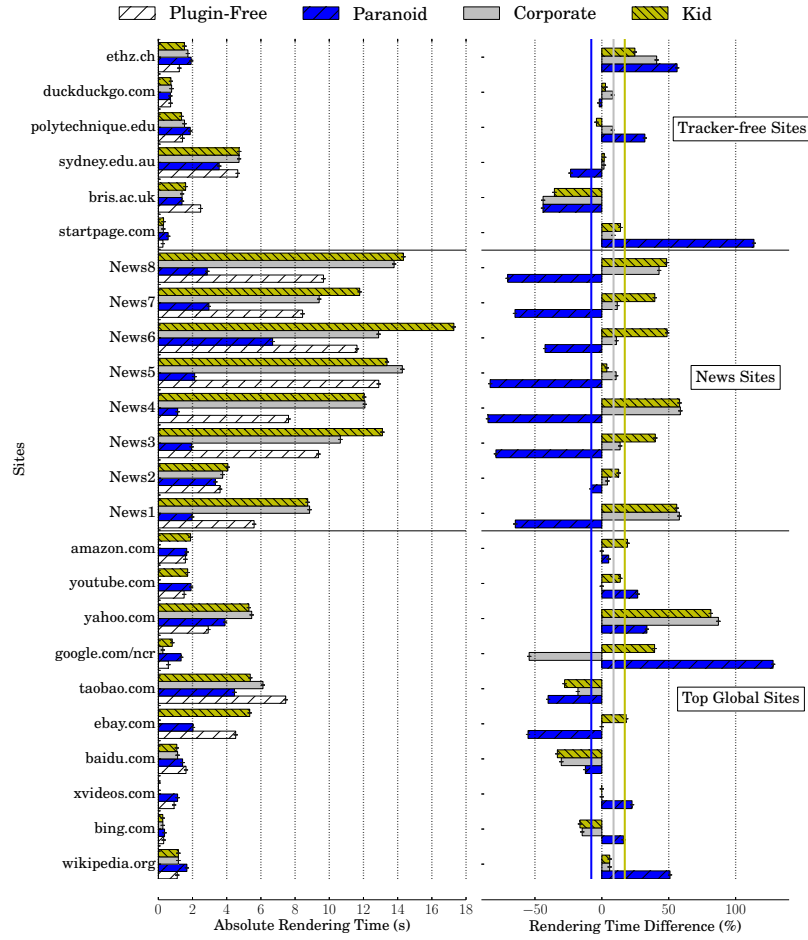


Fig. 3: Page rendering time cost for different plugin setups. Absolute numbers in left-hand plot, relative values with respect to the Plugin-free setup in the right-hand plot.

### 3 Preliminary Prototype

We develop a preliminary CROWDSURF prototype in which we implement the CROWDSURF layer as a Firefox plugin. It supports rules, and the `block`, `redirect`, `log&report` actions.<sup>8</sup> The collector is a Java-based web service, which communicates with clients using SOAP. During the registration phase, the collector provides the CROWDSURF instance with a randomly assigned ID. The collector component receives and stores reports generated by the various CROWDSURF plugins. The CROWDSURF plugin has been installed successfully on both the desktop and the Android version of Firefox. In the following we present simple experiments collected using this prototype.

<sup>8</sup> For instance, the configurations developed for the Corporate and Kid profiles are available at <https://db.tt/yI10LyX1>.



### 3.1 Processing Overhead

First, we evaluate the performance overhead a user would pay when running the CROWDSURF plugin. Given the not-optimized implementation of the prototype, these benchmarks are meant to show the feasibility of the approach rather than being considered as a thorough testing. We consider the three profiles described in Sec. 2.2. For baseline, we take a plugin-free configuration. We setup a testbed based on Selenium WebDriver<sup>9</sup> to automatize the browsing of a selected set of webpages. In particular, we consider i) the Alexa top 10 global websites, ii) 8 news portals, and iii) 6 portals which do not include any online tracker. We run the experiment from a standard PC and instrument the browser to visit each website 20 times. After discarding the best and the worst samples, we measure the average time needed to render the webpage. We purge the browser cache and cookies after each visit.

The left-hand plot in Fig. 3 reports the average rendering time for each website and for each profile. We observe that news portals are the slowest at rendering, most of them taking more than 8 s to fetch and render all the content they embed. Differently, other websites show a much simpler design and their content (mostly being HTML, CSS and Javascript files) are very fast to download. Right plot of Fig. 3 reports the relative average rendering time of each profile with respect to the Plugin-free configuration. For some webpages as *startpage.com* and *wikipedia.org*, the rendering time is very short (order of tens of ms). Thus, the relative difference among the three profiles is broadened, but it is very small in absolute numbers. For the case of *google.com*, the Corporate profile shows much better performance than the Paranoid, since in the former profile requests are redirected to *bing.com*, which in our measurements is faster at rendering. In general, the Paranoid profile is favored, as it blocks advertisement and some Javascript content download, thus speeding up the rendering of the webpage in many cases. The horizontal lines show the average of the relative rendering time for the three profiles. The Paranoid is 1.07 times faster than the baseline. Corporate and Kid configurations show slightly worse performance being 1.08 and 1.17 times slower, respectively.

In summary, results are variable, with more complicated pages that suffer some extra computational costs incurred by CROWDSURF plugin that has to consider and check all links. Nonetheless, being the current implementation not optimized, results hint that clients today have enough power to easily handle the extra load generated by a possible CROWDSURF implementation.

## 4 Motivations for Having CROWDSURF

To demonstrate the need of a system like CROWDSURF, we present some measurement facts. We analyze a 10-day long traffic trace we obtain from a large European ISP collected during October 2014 using Tstat [9]. To analyze both HTTP and HTTPS communications, the dataset includes anonymized TCP logs from more than 19,000 households identified by the modem IP addresses, out of which 11,000 are active (as those IP addresses from which we see at least one HTTPS request and 1000 TCP flows

<sup>9</sup> <http://www.seleniumhq.org>



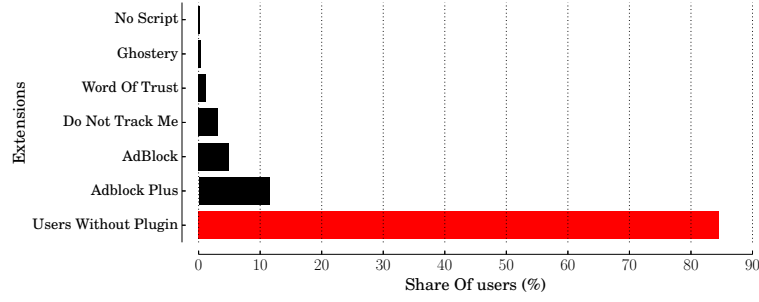


Fig. 4: Shares of users adopting “popular” privacy-preserving extensions.

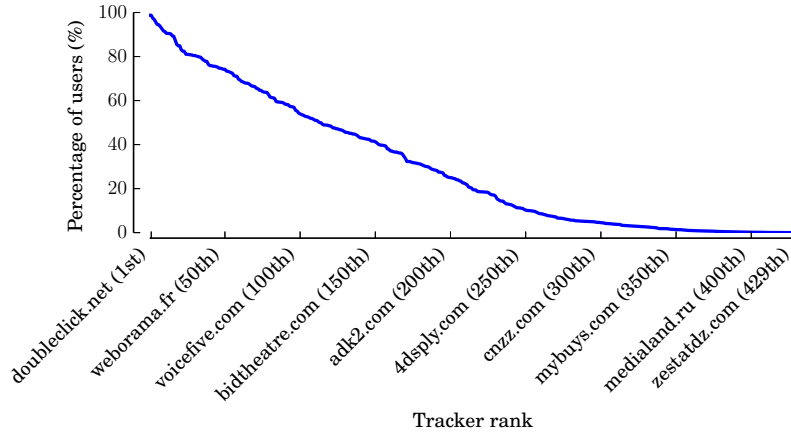


Fig. 5: Percentage of users contacted from top third party tracking services.

in the trace). We leverage DN-Hunter, a technique that allows to annotate TCP flows with original server hostname [6]

#### 4.1 Pervasiveness of Tracking Services

We first observe how many of those users are running any plugin that could help customizing their web browsing privacy. To measure this, for each plugin, we run some active experiments to look for connections toward some update hostname. We then count the fraction of users that contacted such service for updates, and report the results in Fig. 4. The numbers are puzzling: only 3.1% of users have installed DoNotTrackMe, 5% use Adblock, and 11.5% use AdblockPlus. Moreover, more than 84.5% of users do not run any extension to limit advertisement or prevent connections to online trackers. On the other hand, we measure the pervasiveness of most popular online trackers. We build a list of more than 440 tracking services using Ghostery database. We look for users that contact them, i.e., which establish TCP connections toward tracker hostnames. The results illustrated in Fig. 5 are impressive: 98.8%, 98.7% and 97.4% of users regularly (and unintentionally) contact top third party tracking services, i.e., DoubleClick, Google Analytics, and Google Syndication. We count 120 third party services

that are contacted by more than 50% of population. Similarly, 96.6% and 92.4% of users contact Facebook and Twitter (and the tracking services they include) on a daily basis, often involuntarily via the social network buttons embedded on other web pages.

These facts clearly testify how pervasive are tracking services in today Internet, and how users are unaware of their presence.

## 4.2 Checking HTTPS Information Handling

We run a second experiment to verify which data is sent over HTTPS when entering personal information such as user credentials and credit card data on legitimate websites. We collect a dataset by browsing a catalog of websites with a CROWDSURF enabled browser which logs all HTTP and HTTPS requests it observes. We consider a list made of the Alexa top site in the Global, Banking, Gambling and Shopping categories. We investigate a total of 160 top sites. For each website, we manually attempt to log in with the dummy credentials “MyName:MyPassword”. Then, from the collected logs, we check how the client sends those credentials to the servers.

We find that still 10% among the most popular websites in the global rank do not use HTTPS to exchange users’ credentials. Only 2 of these apply some custom encryption/obfuscation technique before transmitting them to the server. Even more surprisingly, among the websites embracing HTTPS in the Global category, we notice that users’ credentials are always sent in plain text over the encrypted channel. Assuming HTTPS offers a secure channel, no guarantees are given on how the server handles and stores credentials. Indeed, the server could store those in plain text, posing severe security risks if the server gets compromised. Unfortunately, this is not a rare event. The most recent incident involved a giant like eBay [4]. Even for the Bank category, 75% of websites transmit credentials in plain text, totally trusting the HTTPS channel. Interesting, some of those do implement two-step strong authentication methods based on pins or tokens, sent in plain text through the HTTPS channel. Similarly, 90% of both Gambling and Shopping categories do not hash the credentials.

These findings strengthen the need for CROWDSURF to warn users about the weaknesses that are unfortunately present on (popular) websites they are used to log in.

## 5 Automatic Detection of Tracker: a Simple Algorithm

One of the design challenges of CROWDSURF is the need of automatic means to detect services that possibly offend users’ privacy. This section presents a simple preliminary solution to automate advice generation. Specifically, we present an unsupervised methodology for an automatic data analyzer to identify possible third party trackers that users unknowingly contact while browsing a given website.

We consider the set of HTTP requests that a user generates when visiting a *target* website. The CROWDSURF layer running in user’s device monitors the HTTP traffic and sends to the collector the anonymized user identifier, and a sample of HTTP request logs having i) the hostname different from *target*, and ii) *target* appearing in the referer field. In other words, CROWDSURF reports to the collector all the third party URLs a user contacts when accessing the webpage *target*. Given this input, the data analyzer looks for parameters in the URLs that may suggest the third party service is using some identifier to track the users.

**Algorithm 1** Automatic third party tracker identifier.

---

```

Input:  $HS, W$  # HTTP request log and target website
Output:  $TS$  # List of possible third party trackers and their user-tracking keys
1:  $H_{ipa} \leftarrow \text{init\_hash\_table}()$  # Init hashtable of IP addresses
2:  $H_{k-v} \leftarrow \text{init\_hash\_table}()$  # Init hashtable of key-value pairs
3: while  $h$  in  $HS$  do # Read HTTP request logs
4:    $h \leftarrow ipa, \text{hostname}, \text{path}, \text{referer}$  # Extract fields of interest
5:   if  $W$  not in  $h.\text{hostname}$  and  $W$  in  $h.\text{referer}$  then # Check target is third party
6:      $K, V \leftarrow \text{extract\_keys}(h.\text{path})$  # Extract keys and values from the path field
7:     while  $k, v$  in  $K, V$  do # Iterate all key names and values
8:        $\text{hostname\_key\_ipa} \leftarrow \text{create\_hash}(h.\text{hostname}, k, h.ipa)$  # Create hash for  $H_{ipa}$ 
9:        $\text{hostname\_key\_value} \leftarrow \text{create\_hash}(h.\text{hostname}, k, v)$  # Create hash for  $H_{k-v}$ 
10:       $\text{ADD\_DISTINCT}(H_{ipa}[\text{hostname\_key\_ipa}], v)$  # Insert all key-value pairs in  $H_{ipa}$ 
11:       $\text{ADD\_DISTINCT}(H_{k-v}[\text{hostname\_key\_value}], h.ipa)$  # Insert the IP address in  $H_{k-v}$ 
12:    end while
13:  end if
14: end while

15: while hash in  $H_{ipa}$  do # Iterate over  $H_{ipa}$ 
16:   while value in  $H_{ipa}[\text{hash}]$  do # Iterate over values mapped to current hash
17:     if  $\text{LEN}(H_{ipa}[\text{hash}]) == 1$  then # Check current hash refers to one value only
18:        $\text{hostname}, \text{key}, ipa \leftarrow \text{decode\_hash}(\text{hash})$  # Decode hash into hostname, key and IP address
19:        $\text{hash}_{aux} \leftarrow \text{create\_hash}(\text{hostname}, \text{key}, \text{value})$  # Create an auxiliary hash using hostname, key and value
20:       if  $\text{LEN}(H_{k-v}[\text{hash}_{aux}]) == 1$  and  $H_{k-v}[\text{hash}_{aux}] == ipa$  then # Check the auxiliary hash in  $H_{k-v}$  contains only one IP address and check this corresponds to the one in  $H_{ipa}$ 
21:          $\text{ADD}(TS, \text{hostname}, \text{key})$  # Add hostname and key to the output list
22:       end if
23:     end if
24:   end while
25: end while

```

---

Our algorithm, illustrated in Alg. 1, extracts all HTTP parameters from the third party URLs. For example, from the third party URL `http://www.acme.com/query?key1=X&key2=Y`, it extracts `key1` and `key2`, with values `X` and `Y`, respectively; `www.acme.com` is the third party hostname. For each hostname and for each key, we investigate one-to-one mapping between the CROWDSURF user identifier and the observed values. Intuitively, we look for keys whose value is uniquely associated to the user. This hints to the key being a “user identifier”, and thus the algorithm labels the third party hostname as “tracker”.

We validate our approach using our passive traffic trace, which contains enough data to pinpoint trackers. We target three popular web portals, News1, YouTube, and Facebook. We check all third party hostnames, and we extract those keys whose values show a one-to-one mapping with the IP addresses of the client which is considered an user identifier in our traces.<sup>10</sup> In this experiment, we consider those keys for which we observe at least 25 distinct IP addresses.

Thus, we run the algorithm to pinpoint the possible third party trackers, and the keys they employ to store the users’ identifiers. Results are reported in Table 2. As shown, we identify 3, 5 and 8 trackers for News1, YouTube and Facebook, respectively. Keys clearly suggest the exchange of possible user identifiers. The only possible false positive is `install_timestamp`, which however we verify manually to be a unique user identifier. Looking at the hostnames, most of them are known tracking services that

<sup>10</sup> The ISP assigns a static IP address to each household modem.

Website	Third party hostname	Keys
News1	<i>pix04.revsci.net</i>	id
	<i>su.addthis.com</i>	guid
	<i>track.adform.net</i>	icid
YouTube	<i>bh.ams.contextweb.com</i>	vgd
	<i>eu-jet-01.sociomantic.com</i>	fpc
	<i>ib.adnxs.com</i>	uuid
	<i>uip.semasio.net</i>	sExtCookieId
	<i>www.wajam.com</i>	install_timestamp
Facebook	<i>adadvisor.net</i>	bk_uuid
	<i>data.bncnt.com</i>	uid
	<i>go.flx1.com</i>	anuid, euid
	<i>ira.spysomeone.com</i>	s
	<i>tags.bluekai.com</i>	google_gid
	<i>ww1.collserve.com</i>	bk_uuid
	<i>www.skyscanner.com</i>	ksh_id

Table 2: The third party trackers and the user-tracking keys we find in our HTTP trace for the targets News1, YouTube and Facebook.

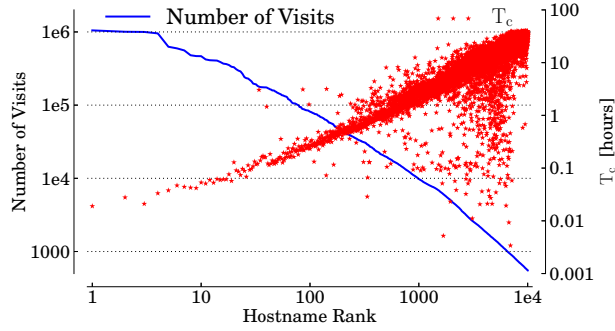


Fig. 6: Popularity of hostnames found in a portion of our trace, and corresponding average time  $T_c$  (in hours) to collect at least 100 with a sample ratio equal to  $1/10$ .

already appear in our list. The only exception is *www.skyscanner.com* (a flight booking website) which tracks the users using the key *ksh\_id*.

Results are promising and show that the availability of large data enables automatic detection of personal leakage information. While the our experiments are carried over HTTP traffic, CROWDSURF allows us to check for privacy leakage also on HTTPS connection, by checking other key-value pair, e.g., in cookies.

## 6 A Feasibility Check

As described in Sec. 2, the crowd feedback is vital for CROWDSURF. Therefore, we study how long the system should take to build a large enough data collection to get reliable analysis results to generate the advices. We consider for instance the case in which we aim at collecting data involving  $N$  different hostnames. We say a hostname data to be reliable when we have collected at least  $K$  entries, and we assume that an en-

try is reported to the collector when a user visits such hostname. We apply the sampling so that only a fraction of entries are eventually seen by CROWDSURF.

Understanding how many visits we need to obtain  $K$  samples for each of the  $N$  hostnames is a problem that belongs to Coupon Collector’s family. In particular, we have to refer to the Newman-Shepp generalization [12]. In this case, the expectation  $E[V]$  of the number of visits  $V$  needed to collect a constant number of entries  $K$  for a large number of hostnames  $N$  is given by:

$$E[V] = N \log N + (K - 1)N \log \log N + O(N). \quad (1)$$

The model assumes visits are equally distributed among  $N$  hostnames. Considering the trace described in Sec. 4 where 19,000 households contacts every day more than 290,000 distinct hostnames. By combining this data with Eq. 1, we obtain that the system would take only 8 days to collect at least  $K=100$  reports for each of the  $N=10,000$  selected hostnames in the catalogue.

In reality, the probability of visiting an hostnames typically follows a heavy-tailed Zipf-like distribution. For instance, on the left y-axis, Fig. 6 reports the number of visits of top 10,000 hostnames. As expected, it follows the typical Zipf-like distribution. Thanks to this, the top 10,000 hostnames correspond to 88.13% of total visits. Therefore, we run a trace-driven experiment using the actual trace to evaluate the average time  $T_c$  needed to collect 100 samples for each of the top 10,000 hostnames in the trace. We assume CROWDSURF clients are configured with sampling ratio equal to  $1/10$ . We focus on the top 10,000 popular hostnames in the first day of our trace. For each request, we measure  $T_c$ , averaging over 12 independent runs, i.e., starting the collection at a random time. As soon as  $K=100$  samples are collected, the hostname advice is said to be reliable.

The right y-axis of Fig. 6 reports  $T_c$  (in hours) needed to reach the minimum critical mass of  $K=100$  visits. As shown, CROWDSURF would take few seconds to collect 100 visits for the most popular hostname (e.g., 87 s for *www.google.com*). Less than 48 h are needed in the worst case. Observe that some services show very bursty traffic patterns that considerably decrease  $T_c$ . Indeed, when the clients access those services, we collect a large number of samples in few time. The overall average value of  $T_c$  is 12.57 h, much less than the time predicted by Eq. 1.

This simple experiment clearly shows that even with a population of only 19,000 contributors that are reporting  $1/10$  of their activity we can easily collect enough data to compute advices. We can also envision smarter sampling policies to, e.g., avoid to keep collecting samples from most popular sites while only asking sample contributions for other services.

## 7 Discussion and Future Work

This paper presented CROWDSURF, a novel crowd-sourced holistic approach to empower informed choices in the web. Motivated by the fact that today service owners have the (almost total) control on information they can collect, and by the fact that users and companies are more and more kept out of the loop, we advocate the need for any user, any device, any network to have the freedom to control the information exchanged on the Internet.

In this paper, we have shown that CROWDSURF is feasible. We presented real data to show how easy would be building a crowd-sourced knowledge, supported by automatic algorithms. As a proof of concept, we implemented CROWDSURF as a Firefox plugin, showing that its benefits can come at a marginal performance cost for the user.

CROWDSURF design presents some practical challenges that must be faced, and ingenuity must be used to find appropriate solutions. The research community as a whole is called to design efficient algorithms, and propose scalable implementations. CROWDSURF offers this possibility, allowing anyone to contribute.

We are aware that our idea is ambitious, as, first, CROWDSURF shall pass through a long and difficult standardization process to get accepted as a compelling technology by the industry, and, second, it shall undergo a deep engineering analysis to convince users about its effectiveness and usability. However, as the community is becoming more and more aware that data constitutes a vital asset in modern web, we are confident that the unified solution offered by CROWDSURF represents a good starting point to protect (and possibly endorse) such asset.

## References

1. Facebook sued for 15\$ billion over alleged privacy infractions,  
<http://www.cnet.com/news/facebook-sued-for-15-billion-over-alleged-privacy-infractions/>
2. Google to pay record 22.5m fine to FTC over Safari tracking,  
<http://www.theguardian.com/technology/2012/jul/10/google-fine-iphone-ipad-privacy>
3. Google SafeBrowsing,  
<https://developers.google.com/safe-browsing/>
4. Hackers steal vast eBay user database, including passwords,  
<http://www.bdlive.co.za/world/americas/2014/05/23/hackers-steal-vast-ebay-user-database-including>
5. Agarwal, Y., Hall, M.: ProtectMyPrivacy: Detecting and Mitigating Privacy Leaks on iOS Devices Using Crowdsourcing. In: ACM MobiSys. (2013)
6. Bermudez, I.N., Mellia, M., Munafo, M.M., Keralapura, R., Nucci, A.: DNS to the Rescue: Discerning Content and Services in a Tangled web. In: ACM IMC. (2012)
7. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: USENIX Security Symposium. (2004)
8. Enck, W., Gilbert, P., Chun, B.G., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.N.: TaintDroid: An Information-flow Tracking System for Realtime Privacy Monitoring on Smartphones. In: USENIX OSDI. (2010)
9. Finamore, A., Mellia, M., Meo, M., Munaf, M.M., Rossi, D.: Experiences of internet traffic monitoring with tstat. IEEE Network (2011)
10. Kramer, A.D.I., Guillory, J.E., Hancock, J.T.: Experimental evidence of massive-scale emotional contagion through social networks. PNAS (2014)
11. Naylor, D., Finamore, A., Leontiadis, I., Grunenberger, Y., Mellia, M., Papagiannaki, K., Steenkiste, P.: The Cost of the “S” in HTTPS. In: ACM CoNEXT. (2014)
12. Newman, D.J.: The double dixie cup problem. American Mathematical Monthly (1960)
13. Popa, L., Ghodsi, A., Stoica, I.: HTTP As the Narrow Waist of the Future Internet. In: ACM HotNets. (2010)
14. Riederer, C., Erramilli, V., Chaintreau, A., Krishnamurthy, B., Rodriguez, P.: For Sale : Your Data: By : You. In: ACM HotNets. (2011)